A Software-Defined IoT Device Management Framework for Edge and Cloud Computing

Alex Mavromatis, *Student Member, IEEE,* Carlos Colman-Meixner, *Member, IEEE,* Aloizio P. Silva, *Member, IEEE,* Xenofon Vasilakos, *Member, IEEE,* Reza Nejabati, *Member, IEEE,* and Dimitra Simeonidou, *Member, IEEE*

***Abstract*—We present the design and implementation of the Software-Defined IoT Management (SDIM) framework based on a Software-Defined Networking (SDN) enabled architecture that is purpose-built for edge computing multi-domain Wireless Sensor Networks (WSNs). This framework can dynamically provision IoT devices to enable machine-to-machine communication as well as continuous operational fault detection for WSNs. Unlike the existing approaches in the literature, SDIM is mainly deployed at Multi-access Edge Computing (MEC) nodes and is integrated with the cloud by aggregating multi-domain topology information. Backed by experimental results over the University of Bristol 5G test network, we demonstrate in practice that our framework outperforms the implementations of the LWM2M and NETCONF Light IoT device management protocols when deployed autonomously at the network edge and/or the cloud. Specifically, SDIM edge deployments can lower average device provisioning time as high as 46% compared to LWM2M and 60.3% compared to NETCONF light. Moreover, it can decrease the average operational fault detection time by approximately 33% compared to LWM2M and roughly 40% compared to NETCONF light. Also, SDIM reduces control operations time up to 27%, posing a powerful feature for use cases with *time-critical* control requirements. Last, SDIM manages to both *reduce CPU consumption* and to have important *energy consumption gains* at the network edge, which can reach as high as 20% during device provisioning and 4.5-4.9% during fault detection compared to the benchmark framework deployments.**

***Index Terms*—Internet of Things, Edge computing, Device Management, Software defined networking, Cloud, 5G**

I. Introduction

THE deployment of massive Internet-of-Things (IoT) infrastructures [1], [2] in urban areas and industrial verticals is in the verge [2], [3], [4], e.g. smart city and industry 4.0. IoT infrastructures are expected to be highly pervasive and composed of millions of heterogeneous physical devices distributed through multiple sites or domains [5], [6], with each domain being connected to application platforms hosted in cloud computing or metropolitan data-centres [7], [8]. Therefore, IoT device management of such a complex infrastructure will be a critical task dealing with important issues of Fault, Configuration, Accountability, Performance, and Security (FCAPS) [9], [6], [10].

Frameworks for IoT device management typically perform device provisioning to initialize machine-to-machine (M2M) communication, the necessary resource control for setting up functional parameters like bandwidth and availability time and, last, device failure detection, IoT network monitoring and mitigation [11]. Most well-known frameworks such as Google IoT Core, Amazon Web Services (AWS) IoT, Azure IoT, or IBM Watson connect a massive number of devices as a single domain and/or in a centralized location by abstracting the underlying infrastructure's network enablers and protocols [6], [11], [12], [13], [14], [15]. Cloud solutions are widely used for IoT, however centralized device management for massive IoT deployments can cause *bottlenecks and large delays.*

As a result, the adopted distribution strategy of IoT platforms between cloud and edge points using Multi-access Edge Computing (MEC) nodes can reduce delay and avoid bottlenecks not only in the data plane, but also in the control or management plane. This is due to the fact that each MEC node can host device management functions and control a subset of IoT devices for a Wireless Sensor Network (WSN) [5], [16]. Nevertheless, there is still room for improving existing single-domain literature solutions on FCAPS [10] regarding multi-domain or multi-edge deployments where several WSNs require management. To this end, Software-Defined Networking (SDN) technologies [17] can enhance device management in multi-domain IoT deployments, as SDN improves control plane [18], device mobility [19] and device resiliency [20], [21], [22].

A. Motivation

In order to cover the above gaps in the literature, the current work studies the problem of multi-domain and multi-edge IoT device management and proposes a *novel* and *robust* Software-Defined IoT Management (SDIM) framework as a solution. SDIM leverages the advantages of SDN to enhance device management functionality and to provide scalable multi-domain IoT deployments, while dealing with FCAPS issues more efficiently. Towards this, our proposed framework is composed of four key novel procedures, described as follows.

1) *Device provisioning:* uses the SDN topology discovery combined to M2M connectivity to enhance provisioning time [23][1].
2) *IoT operational fault detection:* detects operational faults in the IoT field.
3) *Device bandwidth control:* enables different Service-Level Agreements (SLAs) and corresponding Quality of Service (QoS) options over the WSN by modifying bit-rate per device or per WSNs. This allows the business logic to leverage fine-grained details, hence to perform enhanced control over large sensor networks.

[1]This work extends and enhances our prior work in [23].

4) *SDN topology aggregation:* allows to perform monitoring and actions from the cloud to the multiple domains, thus it enables a holistic control and view over all the IoT deployments.

Furthermore, we focus on a smart city use case to demonstrate the feasibility of SDIM. To this end, we present empirical results of a *real* multi-domain IoT deployment in Bristol in the United Kingdom. For comparison purposes, we deploy and test two well-known device management protocols, namely, Light-Weight Machine-to-Machine (LWM2M) [24] and the Network Configuration Protocol (NETCONF) [25]. Our reported experimental results confirm an improvement on the following performance metrics: (i) fault detection time, (ii) device provisioning time, as well as (iii) energy consumption and CPU usage on the MEC side. Finally we conduct experiments focused on SDN topology aggregation time. We also demonstrate how IoT device management performs in the edge and to the cloud.

B. Contribution

Overall, the contributions of our work can be summarized as follows:

- We design and implement a novel IoT device management framework for multi-domain and multi-edge deployments, utilizing SDN principles to provision devices and perform detection of operational failures as well as device control over IoT networks. Moreover, SDIM allows SLAs to leverage fine-grained details and perform enhanced control over large sensor networks.
- We engage into a *real-world* edge IoT field *trial experimentation* over the University of Bristol 5G test network followed by a comparative analysis of SDIM against state-of-the-art protocols, namely, LWM2M and NETCONF Light.
- We present field-trial results for device management performance regarding edge and cloud deployments. Backed by our experimental results, we show significant performance benefits achieved by SDIM compared to these benchmark frameworks. Some highlighted results include that SDIM (i) lowering the average device provisioning times by 60-80% compared to NETCONF Light and 46-60.3% compared to LWM2M. (ii) reduce the average operational fault detection time by approximately 33% and 40% compared to LWM2M and NETCONF, respectively; (iii) reduce control operations time from 23% up to 27% compared to both benchmark frameworks, thus matching the needs of time-critical use cases such as eHealth or safety in industrial environments; and last, (iv) reduce CPU and energy consumption at the edge during both device provisioning and operational fault detection compared to the other benchmark deployments. Note that we also present results for SDN topology aggregation time and overall multi-domain WSN management performance.

C. Paper structure

This article is organized as follows. Section II introduces the background concepts and the related work in the literature by providing a short survey related to IoT device management. Section III presents our proposed framework architecture and work-flow. Section IV presents a detailed description of our test-bed and experimental validation setup, before proceeding with a meticulous performance assessment of SDIM compared to state-of-the-art benchmark models in Section V. We conclude our work and discuss our future work goals in Section VI.

II. Background and Related Work

Before we introduce our framework, we present the background and related works on multi-domain IoT architecture and management followed by its evolution towards Software Define Networking (SDN) IoT.

A. Multi-domain IoT Architecture

The Information and Communication Technology (ICT) infrastructure that supports IoT applications is formed by cloud/edge data-centres connected to networks comprised by millions of heterogeneous IoT devices [4], [5]. A group of IoT devices provisioned for a specific application defines a logical domain associated to a physical domain defined at the edge e.g., Wireless Sensor Network (WSN) [6]. A Multi-access Edge Computing (MEC) node will play an important role for edge network services of IoT physical domains [26].

To understand the interaction between elements of such a complex infrastructure, we refer to the five-layered paradigm architecture presented in [6], [27]. The two higher layers focus more on the interaction between the applications hosted to the cloud and multiple logical and physical domains. The lower layers focus on the interaction inside the domains (e.g., MEC node and IoT devices). The proposed upper layers are:

- **A Business layer,** which abstracts the IoT infrastructure from the main software platform hosted in cloud data-centres (e.g., Google IoT Core, AWS IoT, Azure IoT, IBM Watson etc). Typically, the main software includes the IoT management and control platform, the core applications, the data analytic software, and big data [2], [16].
- **An Application layer,** which defines tools and protocols serving the applications and platforms from the business layer (e.g., Constrained Application Protocol (CoAP) [15], Message Queuing Telemetry Transport (MQTT) [28], HyperText Transfer Protocol Representational State Transfer (HTTP-Rest) [29]).

The proposed three lower layers are:

- **A Service-management layer,** that focuses on the middleware and virtualization enablers supporting the interaction between applications to the cloud and devices at the edge. In this layer, virtual and physical network functions and middleware protocols are deployed on top of existing ICT infrastructures [8], [30]. SDN controllers work in this layer to interact with IoT agents and resource brokers and orchestrators through network protocols [30].
- **An Objects abstraction layer,** which includes network devices and protocols, and compute nodes deployed

to support IoT devices. The MEC node hosts the components of the service management layer by enabling protocols and frameworks to connect and manage IoT devices. Some protocols and frameworks used for IoT device connections are 802.11x, Long-Term Evolution (LTE), Ethernet, UDP, TCP, and SDN. Some device management protocols are IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) [28], Network Configuration (NETCONF) protocol [25], and Light-Weight Machine to Machine Management (LWM2M) [24]).

- **Objects layer** also known as **perception layer,** that focuses on the IoT device and its physical environment. All physical connections and data digitization are performed in this layer. Some basic device detection or management also can be done in coordination with the upper layers. WSNs are defined in the object layer [2], [3].

The IoT platform for management and control and the applications are moving gradually from the cloud to the edge, in order to be closer to the users and devices [16]. This is why in our model we define a domain as a set of devices connected to one MEC node or edge node. A domain is defined between service management layer and abstraction layer where the MEC acts as egde points-of-presence (E-PoPs). Infrastructure and Device management in multi-domain IoT infrastructure is a complex task [11], and a main focus of our study. Therefore, and in order to provide a better understanding of the problem, we provide a brief survey of the main concepts and frameworks related to IoT management from the perspective of infrastructure and the devices in Section II.B.

B. IoT Management Protocols

The Internet Engineering Task Force (IETF) and International Telecommunication Union (ITU) categorize IoT infrastructure management frameworks based on their abstraction (i) Management of ICT infrastructure, and (ii) IoT device management. [11]

1) Management of ICT infrastructure for IoT: This management part focuses on the infrastructure that supports IoT devices by performing three main tasks [11], [27]: (i) Provisioning of the devices. (ii) Monitoring or permanent observations to detect any abnormal operation (e.g., failures). (iii) Administration of infrastructure including the planing, diagnosing, and debugging of the whole infrastructure. Given their high complexity, the management tasks can be divided in application level, system level, and network level ones [27]. At the application, the management is performed inside each application. At the system level, the management is done inside and between interconnected operating systems. Finally, at the network level, the management is performed over network elements such routers, servers, switches, controllers, etc.

2) Device management: Management of many devices usually is focused [11] on provisioning, monitoring, and debugging [12], [27]. IoT deployments include devices with heterogeneous and/or constrained compute, storage, and connectivity capabilities. As a result, device interoperability, scalability, are serious issues for device management [3], [16], [5], [31]. To deal with them, protocols and frameworks were proposed and extended by researchers and international communities for standardization (i.e., the Open Mobile Alliance (OMA), the Broad Band Forum (BBF), the Internet Engineering Task Force (IETF), and the International Standard Organization (ISO)).

Two widely used application layer protocols for IoT device management are the IETF Constrained Application Protocol (CoAP) [15], [31], and the ISO Machine Queue Telemetry Transport protocol (MQTT) [15]. For device monitoring purposes, CoAP adds a unified Resource Identifier (URI) to each IoT device and send regular pings to check their availability. MQTT uses a centralized broker to manage multiple M2M communications over TCP. The first protocol to complement CoAP for device management is the OMA Device Management (OMA-DM) [13] which deploys a bootstrap server for device provisioning, monitoring, and debugging. The server manages parameters of each device such as an associated proxy server, wireless access points, and access rules to provide basic trust (security).

CPE WAN Management Protocol (CWMP-TR-069) [32] allows the management of large number of devices interconnected by a Wide Area Network (WAN) composed by modems, routers, gateways, and devices. Another similar concept for message-oriented IoT architecture was studied in [15] as an extension of the ISO Machine Queue Telemetry Transport protocol (MQTT) based architecture to improve the M2M IoT device management. Two private initiatives for enhance IoT device management are the Wireless Machine-to-Machine Protocol (WM2MP) [33] and the iDigi Device Cloud [34].

The OMA Light-Weight Machine-to-Machine Management (OMA-LWM2M) [14] was proposed as an extension of OMA-DM. OMA-LWM2M provides an agnostic interface between physical devices and servers for remote management [13]. In this work, we deploy and validate our SDIM framework by comparing its permanence with NETCONF light and LWM2M protocols. IETF introduced the Network Configuration (NETCONF) Light protocol to extend the Simple Network Management Protocol (SNMP) in order to manage heterogeneous devices [25], [35]. As benchmarks for comparison we select LWM2M and NETCONF Light, the main reason we select these is that they are lightweight and they are some of the latest implementations for device management. Below we list and survey advantages and weaknesses of both approaches.

C. LWM2M and NETCONF Light

LWM2M and NETCONF Light are platforms posing similarities, built for network device management. Both of them are widely used and investigated in order to accommodate future massive IoT deployments. LWM2M is a UDP-based protocol that is particularly built for constrained devices, hence using primarily CoAP.

LWM2M communication is done via SMS or UDP messaging, resulting to some level of a device-agnostic

protocol. LWM2M is fairly secure using Datagram Transport Layer Security (DTLS) and operates with pre-existing data models. These data models are mutable and extendable, thus enabling multiple use cases. However they are not sufficient for enterprise devices [32] to perform management on the fly. Future extensions will also look at integrating TCP at LWM2M, however at the moment one of LWM2M shortcomings is UDP and the data reliability is quite low in high bandwidth applications. Additionally, CoAP utilization results to NAT issues which is a blocker in NAT networking applications.

NETCONF Light [25] is an IETF network management protocol published which is a lightweight design to target constrained IoT and other applications. NETCONF Light is simply a lighter version of NETCONF that was introduced in RFC 4741, the protocol data modeling language is YANG and coupled with NETCONF software is extremely powerful. The virtues of NETCONF are the extendability and the plethora of options when it comes to configurations. Additionally NETCONF is aligned with SDN principles which is used for management of network elements divided over control and data plane. In terms of IoT NETCONF Light can be a good option, however the pluralism of abilities comes with a complexity penalty, thus not suitable for many heterogeneous constrained devices. Another incompatibility with IoT deployments that NETCONF faces is the session oriented approach, this is not regulated for low-power IoT networks where sleep and wake up functions are main part of their architecture.

Our framework exploits the usage of MEC nodes utilizing SDN. Hence, in this work we study, test, and propose a framework based on the integration between infrastructure and device management for complex multi-domain IoT deployments. The main motivation is born from the centralized monitoring or debugging tasks in large and dynamic IoT deployments that is not scalable [36], [37]. Additionally, our research shows that SDN can work well with heterogeneous IoT devices. Before introducing our framework, we revise related works on SDN in MEC IoT infrastructure in the context of control and device management.

D. Related Work

Software Defined Networking (SDN) technology can provide flexibility, security and efficient resource sharing in IoT infrastructures. Flexibility is required by many applications in terms of inter-domain mobility, M2M communications and addressing. SDN controllers provide a unified view of the network, as a result it can be deployed to provide device mobility [19], adaptable M2M communication [38], and device address management [39]. Hence, infrastructure and device management can be simplified by adding an SDN controller. One key challenge for massive IoT infrastructures is how to secure data routing and device provisioning [5]. [40] studies an SDN architecture for securing data routing by deploying a centralized verification mechanism inside the SDN controller.

The SDN controller plays an important role for network resource sharing or slicing and overall efficient use of network resources [41]. In [42] SDN controllers are deployed in the IoT infrastructure to enable network sharing between smart home applications. However, a centralized SDN controller introduces the risk of a single point of failure or attack [18], [39]. In addition, large number and diversity of IoT devices in terms of capabilities and application requirements limits the scalability and performance of a centralized control. Therefore, multiple controllers needs to be deployed to provide resiliency and to deal with scalability and performance issues [19], [43]. Authors in [17] proposed a comprehensive cloud oriented SDN framework for IoT deployments (SDIoT). This framework deploys software-defined security, routing and control (e.g., device management) for large IoT deployments. However, it does not consider multi-edge deployments to deal with the multi-domain IoT device management problem, which is the key focus of our model.

The placement of multiple SDN controllers in clusters can be performed dynamically or statically to minimize latency and resource usage [44] or maximize efficiency [17]. However, time-critical IoT applications demand millions of flows and routes as well as extremely low latency [45]. To deal with this scenario [46] and [47] proposed an approach for optimal placement of SDN controllers based on the number of flows and routes. As a result, device and network management of IoT infrastructures combined with the placement of multiple SDN controllers can provide load balancing, fine-grained traffic forwarding, and improve the bandwidth availability [47]. Another example of utilizing SDN at the edge is introduced in [48]. This work focuses on the usage of SDN technology for the mitigation of security attacks on IoT devices.

It is also important to focus on automation when SDN is used for IoT networks. In [49] and [41] the authors implement a mechanism that automates services utilizing SDN capabilities. Indeed SDN can improve automation especially in the edge. The authors also focus on cloud performance to prove that the edge nodes can be utilized quite successfully with SDN. In this work we study a further design of edge and cloud SDN and present an analysis on our findings. In Shafi et all [50] the concept of edge SDN-based anomaly detection is introduced, the authors use machine learning. This work presents promising results for the detection of attacks, however, there is no focus on infrastructure operational faults.

Our framework considers a set or a cluster of SDN controllers per instance of IoT infrastructure management. This set is composed by a Master SDN controller placed at the cloud and at least one edge SDN controller placed in each domain or edge locations. Therefore, this integration can remove single-points of failure and introduce localized/targeted device management at the edge, which improves flexibility, scalability, mobility, and resiliency of IoT deployments.

In the following section we introduce the IoT management problem and our proposed framework.

III. Software Defined IoT Device Management

A. IoT Device Management Problem

IoT platforms will face scalability issues with unprecedented amount of devices connected to the internet.

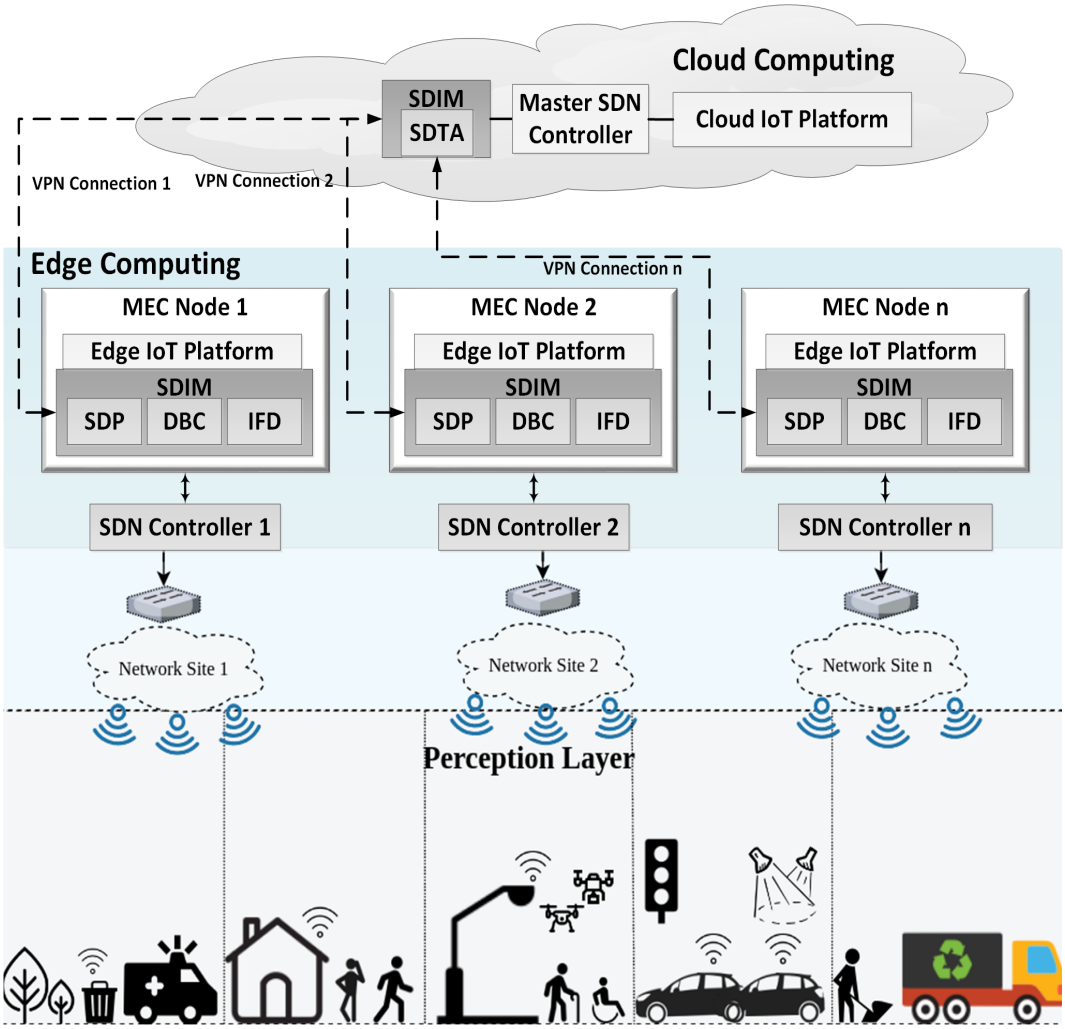


Fig. 1: Architecture of SDIM Framework for Edge and Cloud IoT Infrastructure

To this end, edge computing emerges as solution [51], [35] for some of them i.e., real time applications requiring IoT data. Managing all those devices to the cloud will produce application and network bottlenecks. As a result, the investigation of new possible ways for utilizing the advantages of multiple edge infrastructures (e.g., multiple domains) is essential, not only for data processing but also for decision making.

The problem of device management is typically found for smart city cases because the infrastructure must accommodate resources for different verticals, e.g. Vehicle-to-Vehicle (V2V) communication, emergency networks, and many others IoT networks running in parallel. Therefore, in this paper we study the device management scalability issue for a smart city case by deploying and validating a new device management framework.

Our framework is primarily based on the edge computing paradigm and SDN technology. More precisely, we leverage on lightweight device management services that can be implemented as containers on very constrained MEC devices, hence *eliminating* the handshake and control signals exchange with the cloud. This results to faster response times compared to the centralized device management. Therefore, by multi-domain device management the task becomes less complicated, and the idea of device management decentralization leads us to utilize SDN to its fullest extend. SDN can provide the global view of networks, as a result provide multi-domain information that allows multi-domain monitoring and control. Given that SDIM is designed to solve *logically* centralized bottlenecks and utilize both edge and cloud. In the following subsection we present in detail the SDIM framework architecture.

B. *Software-Defined IoT Management Architecture*

The SDIM architecture is based on a multi-domain network setup, where each domain is integrated with layer two and layer three SDN. By exploiting SDN topology detection and flow control over each network site, SDIM is able to construct global networks view, and at the same time isolate the device management procedures at the MEC nodes. To this end, SDIM architecture can be divided into four main building blocks, where three of them are deployed at the MEC nodes and one is deployed to the cloud. SDIM overall architecture is presented

in Figure. 1, and the three edge building blocks are described below:

1) *Software Defined Provisioning (SDP)* is the element that initializes any communication between the devices and the IoT platform. Based on a white list of MAC addresses, SDP is able to cross-validate the white list with the SDN topology, providing zero touch device provisioning.
2) *Device Bandwidth Control (DBC)* monitors the bytes of incoming packets and applies rules concerning the maximum allowed bandwidth. DBC is the element that reads Service Level Agreements (SLAs) and applies them over a specific number of devices. Therefore, control signals for enabling or disabling devices are also part of DBC. The business logic integrates at this point, by deploying multiple descriptors describing SLA's with different requirements that are mutable over time.
3) *IoT Fault Detection (IFD)* detects failures in M2M communications by validating in real time the MAC address of each frame with a white list. IFD is able to detect two types of operational failures: (i) Unavailability of data due to an inactive device. (ii) Unauthorized devices to avoid intrusions. IFD does not focus on application security faults.

The aforementioned architecture elements are placed on each MEC node and are connected to the cloud elements through a dedicated Virtual Private Network (VPN) link. The VPN connectivity is used for control plane traffic of SDIM and for data traffic generated by IoT platforms and applications (e.g., big data). To the cloud, SDIM deploys two elements:

1) *SDN Topology Aggregation (SDTA)* that provides a global view of the IoT deployment to a Master SDN controller, by aggregating all the topology information for each of the domains.
2) *The Master SDN Controller* that uses the aggregated topology information from the SDTA to enable Cloud SDN.

In the next subsection we describe the SDIM system overview.

C. SDIM Framework Overview

SDIM framework overview is divided into four main procedures (Wireless SDN, Software Defined Provisioning, SDN Topology Aggregation and IoT operational faults detection). The procedures of the framework are interacting with all the network elements that comprise IoT end-to-end applications, such as devices, Access Points (APs), the SDN controller which is a requirement of SDIM (both domain and cloud), and the IoT platforms. In Figure. 2 a sequence diagram of the SDIM system is introduced, and all SDIM procedures described in detail below:

1) Procedure one: Wireless SDN: SDIM matches rules for the devices based on source MAC addresses and the IoT platform destination IP address. To this end, SDIM SDP component requests the topology of devices connected to the AP from the AP's Application Programming Interfaces (APIs). Using the AP API SDP is able to update the SDN controller LLDP inventory and recreate the SDN topology. In this way, the SDN controller holds the WSN topology information and is able to add flow rules over each device or device groups. This process is based on MAC and IP address filtering, which is more efficient compared to IP layer control messaging (as verified by our evaluation results), especially for larger-scale sensor networks. Utilizing LLDP reduces significantly the computational requirements and time for operational fault detection, device provisioning and device control.

2) Procedure two: Software Device Provisioning (SDP): The device provisioning is based on two main loops presented in Figure. 2. The first loop is defined as provisioning Loop (pL). Considering that the WSN topology is integrated at the SDN controller, SDP is able to operate the provisioning. Provisioning happens by cross validating the MAC addresses currently connected at the APs with a white-list of MAC addresses, this process runs again if the WSN topology is updated.

The cross validation is implemented based on the Algorithm 1. Algorithm 1 is utilizing binary search which has a known time complexity of $\mathcal{O}(\log n)$, where n is defined as the number of devices to be provisioned. Furthermore, a for-loop is used to assess the status of each device. The for loop has a time complexity of $\mathcal{O}(w)$ where w represents the elements of the white list. Considering the binary search is executed as many times the algorithm checks for the device status, the total complexity of the Algorithm 1 is: $\mathcal{O}(\log n)$ * $\mathcal{O}(w)$. This results to $\mathcal{O}(n * \log w)$ time complexity for Algorithm 1.

3) Procedure three: SDN Topology Aggregation (SDTA): SDTA constructs the global multi-domain network view by integrating the topology inventory of all SDN controllers. During the setup period, each SDN controller of the WSNs subscribes to the cloud master SDN controller and the inventory aggregation is automatically updated. Therefore, the master controller gathers all the topology information, which allows cloud SDN. SDTA is dynamic and incorporates new network topologies on the fly.

4) Procedure four: IoT Fault Detection (IFD): IFD component is depicted in Figure 2 as main loop faults Loop (fL). This procedure is implemented as an event that monitors incoming packets from the SDN switches (packet ins) and targets operational faults. Additionally, to enhance the fault detection procedure we develop also the Device Bandwidth Control (DBC) component. DBC is implemented by monitoring the byte count of each device and accordingly adds a flow rule to the switch.

Furthermore, IFD implements Algorithm 2, the time complexity is not very different to Algorithm 1, due to the similar binary search use. However in the fault detection case the algorithm will check packet in messages from the SDN controller. To this end the for loop complexity will be: $\mathcal{O}(p)$ where p is given by the following: $p = Df * N$. So for the algorithm worst case time complexity we have: $\mathcal{O}(p)$ * $\mathcal{O}(\log w)$ resulting in $\mathcal{O}(p * \log w)$. Details about implementation and field trial experimentation can be found in Section IV.

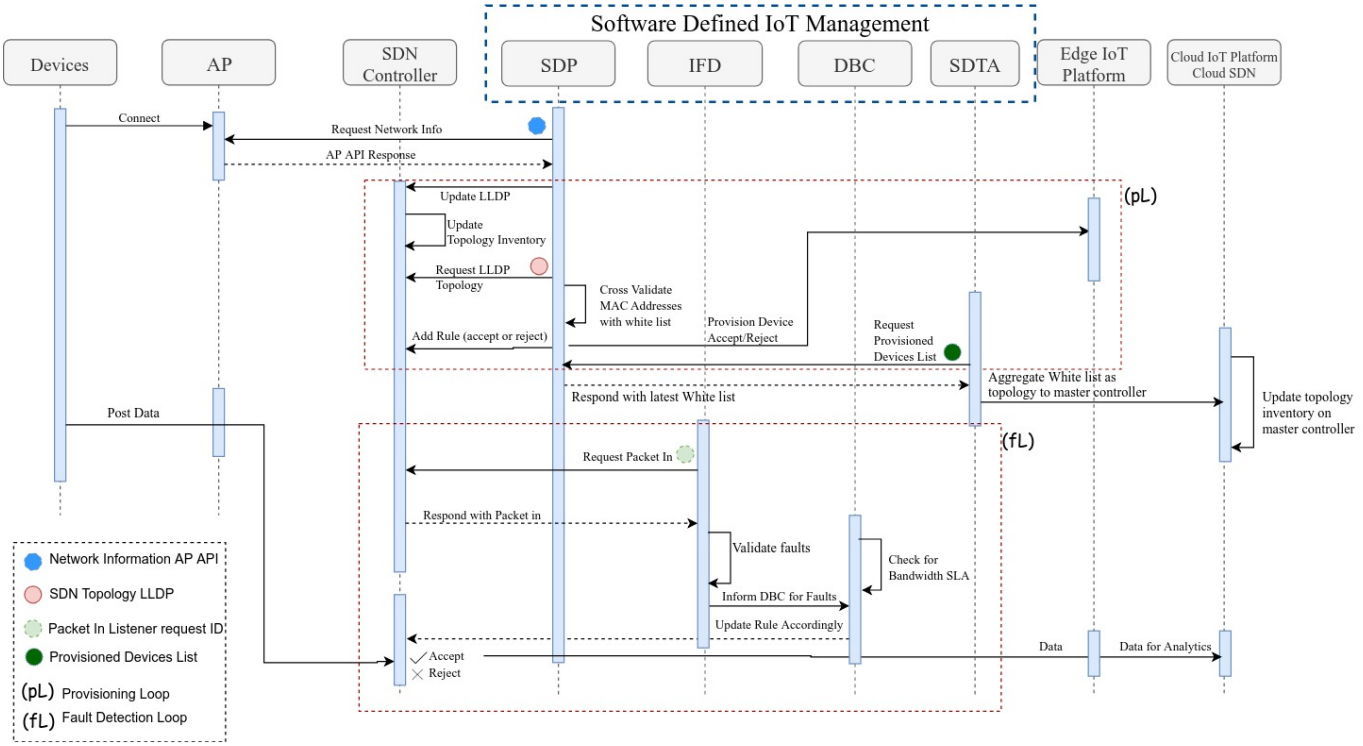


Fig. 2: SDIM System Overview

IV. Experimentation Setup

A. Benchmarks for Comparison with SDIM

To perform a concrete performance evaluation of the proposed SDIM framework, we conduct a comparative evaluation between the state of the art IoT device management protocols and our proposed SDIM. We elaborate further on the choice of the protocols for comparative evaluation in related work Section II. To smoothly usher into the results we briefly introduce the way each of our benchmark frameworks operates.

- LWM2M: specifies four stages that operate the device provisioning and M2M communication. The first stage is called bootstrap interface that allows devices to collect the crucial information for communication establishment. For device provisioning a registration interface is used, where the client (device) sends a message to establish the destination that wants to communicate with (i.e IoT Platform). This is a continuous procedure where the device needs to be synchronized for keeping the registered state on. The third interface is the device and service management enablement. At this stage the server that runs LWM2M already has information about the devices and in that way interacts with devices. Finally, there is a fourth interface called information interface where LWM2M is able to observe the resources where devices are sending periodically notifications. This interface is used for fault management.
- NETCONF Light: is comprised of four main layers and multiple operations. However NETCONF light implements only the necessary operations for each application. The operations are getting and setting configurations across the registered devices and they are described as YANG service and device models. The main idea behind NETCONF light is that devices need to support at least one session which is being used for communication. Finally, interaction with devices are done by using session-specific attributes. The NETCONF protocol provides a small set of low-level operations through which the device monitoring and fault detection is enabled. The fault detection reports inactivity timeout, or detection of unusual behavior from the devices.

Our proposed framework targets to surpass the aforementioned device management approaches, in Key Performance Indicators (KPIs) defined in Section V. All of the experiments conducted for this paper are performed over the University of Bristol 5G Test Network (UoBTN) which is described in the following subsection.

B. MEC IoT Test-bed

For the evaluation of the proposed framework, we have deployed a sensor network integrated with MEC capabilities over the UoBTN. As shown in Figure 3 the UoBTN is located in Millennium Square at the Bristol city centre. For this work we have extended the test-bed with installing five MEC nodes and twenty sensor devices. The following three subsections are describing in detail, the network topology, the IoT and MEC node integration and setup and the implementation software. Explicit details about software and hardware are reported in Table 1.

1) Network Topology: The network topology used for experimentation and validation is presented in Figure 4. To

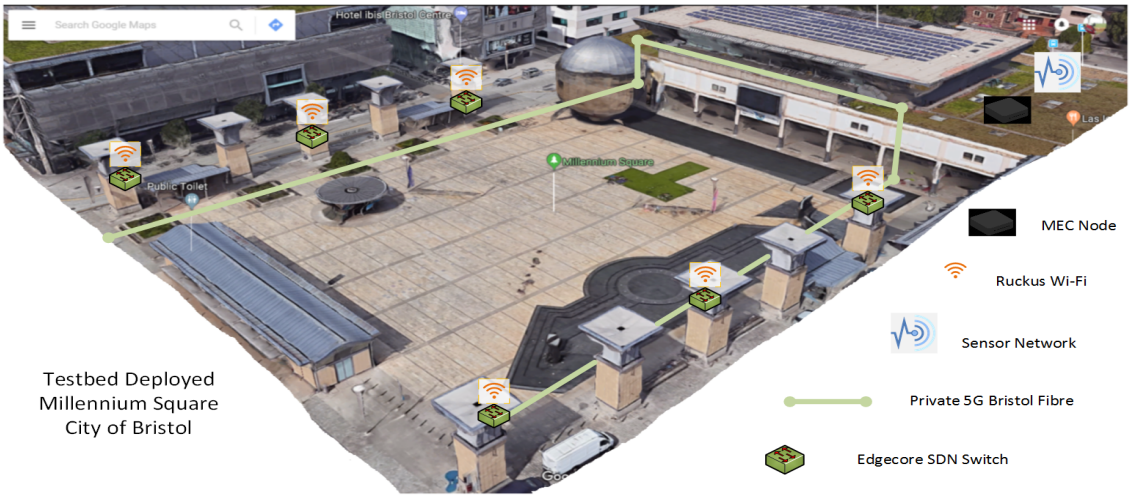


Fig. 3: Bristol Smart City Test Bed at Millennium Square

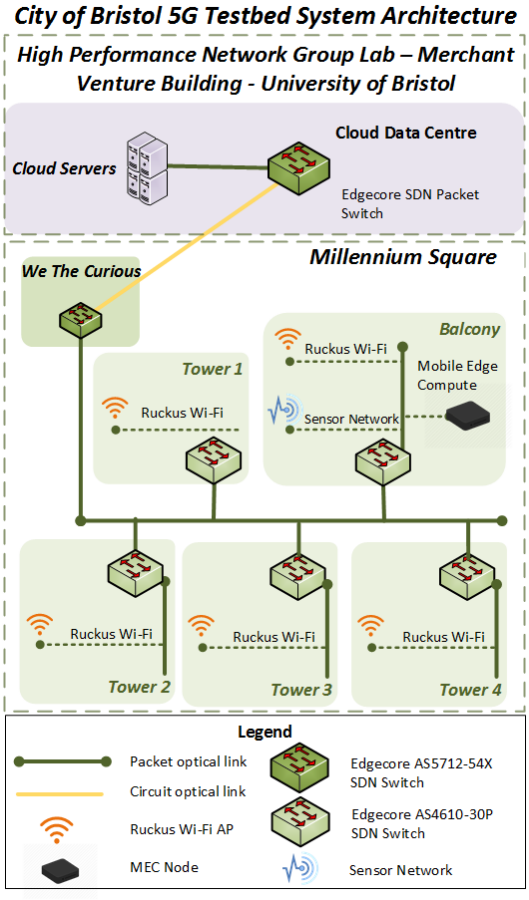
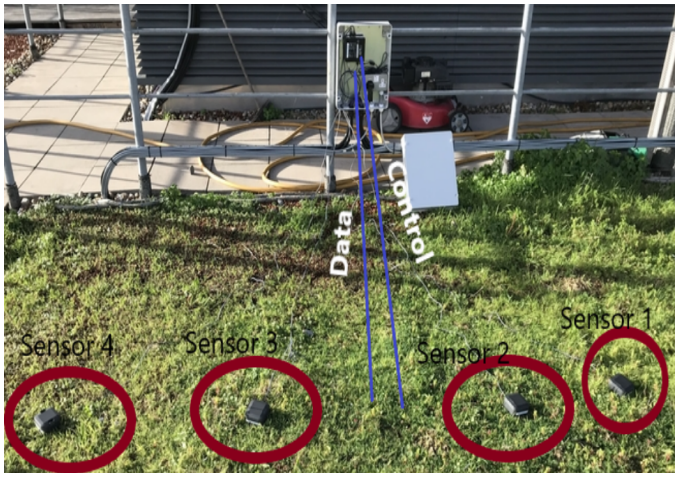
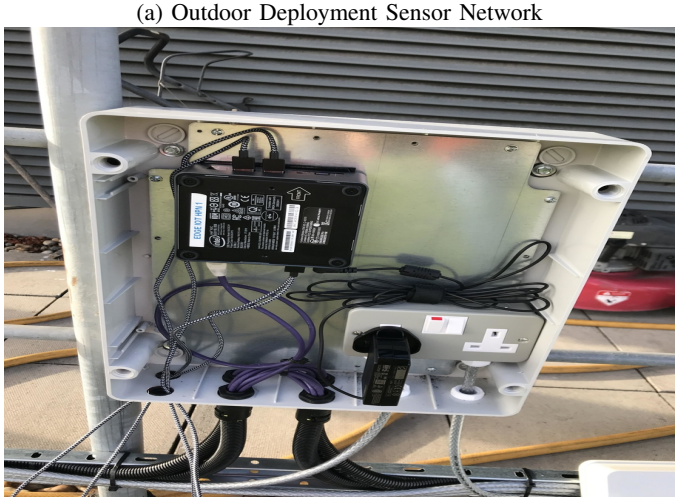


Fig. 4: University of Bristol 5G Test Network Topology for Experimentation



(a) Outdoor Deployment Sensor Network



(b) MEC Node Outdoor deployment

Fig. 5: Device Provisioning Comparison of Field Trial and Emulated Large IoT Networks at the Edge

enable the multi-edge and multi-domain WSN setup, we connect a group of devices with multiple WiFi APs. Each of these APs are integrated with an SDN switch where in the same sub-net we are also connecting the MEC nodes. As is highlighted in Figure 4 a fiber link is connecting all the wireless and edge compute nodes with the High

Algorithm 1: Device Provisioning/Blacklisting Algorithm

Data: SDN Topology S, White list W, Provisioned Devices P, OpenFlow Rules SDN, IoT Platform Address IIP, Black List B

while *! S.updated()* **do**
  If SDN topology is updated start again
  $W \leftarrow getwhitelist()$;
  foreach $device \in S$ **do**
    if
    $(BinarySearch(device.MAC \notin P.MAC)$
    AND
    $(BinarySearch(device.MAC \in W.MAC)$
    then
      $P \leftarrow$device.MAC;
      SDN.addrule(Accept, $IIP \leftarrow$device.MAC);
      $W.MAC.remove(device.MAC)$;
      $P.MAC.remove(device.MAC)$;
      remove MAC address until topology gets updated
    else
      $B \leftarrow$device.MAC;
      SDN.addrule(Accept, $IIP \leftarrow$device.MAC); black listing the device using SDN
    end
  end
end

Algorithm 2: Faults Detection Algorithm

Data: SDN Topology S, White list W, Provisioned Devices P, OpenFlow Rules Function SDN, IoT Platform IP Address IIP, Black List B, Detection Frequency Df

$ExpectedDevices \leftarrow emptylist()$;
creating expected devices list for cross validation
$Nsize \leftarrow P.size()$;
while *Df.NotChanged()* **do**
  If SDN topology is updated then start again
  for $i \in [0, |Nsize|]$ **do**
    $Fault \leftarrow$BinarySearch($SDN[i].MAC, P[i].MAC$);
    if $Fault$ **then**
      $ExpectedDevices[i].MAC \leftarrow P[i].MAC$
    else
      Fault Detected $B \leftarrow$SDN[i].MAC;
      SDN.addrule(Reject,$IIP \leftarrow$SDN[i].MAC)
    end
  end
  if $ExpectedDevices.Size() \neq P.Size()$ **then**
    $ReportFaults(ExpectedDevices)$
  end
end

Performance Networks (HPN) group data centre facilities. It is important to note, that for this paper we have used AWS for our cloud experiments. AWS is used as cloud because the fiber connectivity between the HPN lab and the edge network provides latency in terms of nanoseconds, therefore for realistic cloud experiments we have deployed all the software at AWS.

2) Edge Nodes and IoT Integration: In order to evaluate the benefits of SDIM, as well as the IoT device management at the edge of the network, we have deployed a sensor network. 20 environmental Pycom Pysense sensing devices are installed as depicted in Figure 5a integrated with the MEC nodes that are connected to the SDN switches. An example of one out of five total MEC nodes setup is illustrated in Figure 5b. The MEC nodes are in the same sub-network of four devices to enable communication for provisioning and operational faults detection. In the next subsection the software setup of the MEC nodes is explained, as well as any other additional software installed.

3) Software Implementation: Most of the software used in our experiments is deployed on the MEC nodes. The MEC nodes in our setup are Intel NUCs and we run all the software on docker containers. SDIM is developed for this research and it is written with Node js and python. For the LWM2M we use the open source implementation of Eclipse Leshan. Leshan is an OMA Lightweight M2M (LWM2M) implementation in Java which enables us to experiment with. For NETCONF we use the OpenDayLight (ODL) implementation in Java, where we extracted only the components specified in the NETCONF light. Furthermore, as SDN controller we work with ODL and OpenFlow1.5, and SDIM is utilizing the REST API of ODL. It is important to note that the software environment and computation resources are the same in all experiments, in order to assure consistent results.

For the emulation of devices we use Mininet WiFi [52]. In this way we can emulate the SDN connectivity of a larger amount of devices, without having any significant deviation from the real scenarios. The wireless SDN is implemented with the integration of Ruckus WiFi API and the ODL LLDP topology inventory. Finally, both in the real sensor case and the emulated case we implement MQTT protocol to transmit data. The traffic rate follows Poisson distribution with range 10 to 20 messages per minute, and the message size is 500bytes. We select these values to create a scenario of high traffic for IoT applications.

V. Results and Discussion

This section presents a comprehensive performance evaluation analysis between SDIM and two existing approaches, namely LWM2M and NETCONF, in both edge and cloud computing scenarios. Our head-to-head comparison is based on eight different key metrics defined bellow.

1) **Average Provisioning Time**: We calculate the difference between the time of the device connection established with the AP, with the reception time of the first packet at the IoT platform. It is important to note that the figures plotting device provisioning evaluation are presenting the performance of all approaches after the initialization period. The initialization period is the

TABLE I: Emulation & Experimental Setup

	Emulation	Experiments
Experimentation Details		
Number of Devices	5000	20
Duration	Random	Random
Traffic Rate	Poisson[10-20]msg/min	Poisson[10-20]msg/min
Fault Rate	Poisson[40-80]msg/min	Poisson[40-80]msg/min
Message size	$500bytes$	$500bytes$
Protocols		
Messaging	MQTT	MQTT
Transport Layer	TCP	TCP
WiFi	802.11ac	802.11ac
SDN Discovery	LLDP	LLDP
SDN Flows	Openflow 1.5	Openflow 1.5
Hardware & Software Specifications		
Sensors	Mininet WiFi	Pycom Pysense
MEC	Intel NUC	Intel NUC
Total MECs	5	5
CPU	Intel Celeron 1.50GHz	Intel Celeron 1.50GHz
RAM Capacity	2GB	2GB
Hard Drive	32GB	32GB
WiFi Access Point	R610	T610
SDN L2 Switch	Edgecore AS4610-30P	Edgecore AS4610-30P

time required by each approach to acquire all required information for provisioning, such as predefined IP addresses, ssh keys and SDN topology information. The average provisioning time for the IoT devices required is defined by Equation (1):

$$\bar{\tau}_{prov} = \frac{1}{n} * \sum_{i=1}^{n} (\tau p_i - \tau c_i) \tag{1}$$

Where $\bar{\tau}_{prov}$ is the average provisioning time, n is the total number of devices, τp_i is the time where i is the ith device that has been provisioned. Finally τc_i denotes the time that each device connected to an AP.

2) **Average Fault Detection Time**: We implement a mechanism of inserting manually operational faults in the IoT field. The faults are of two types: The first type is disabling randomly devices in the field. And the second type is adding blacklisted or new devices to the field, that are trying to sent data. With these actions, we are able to bring all the approaches to their limits of failing to detect 100% of the aforementioned faults. It is important to note that SDIM, LWM2M and NETCONF Light have diverse abilities of detecting different faults due to the different architectures. In order to perform a fair comparative analysis we only focus to the aforementioned two types of operational faults for all approaches. And evaluate realistic operational anomalies of IoT deployments. The fault insertion to the field is following a Poisson distribution of 40 to 80 faults per second, which is an estimated value for the ultimate braking point of all the approaches. The average fault detection time is calculated by:

$$\bar{\tau}_{fd} = \frac{1}{n} * \sum_{j=1}^{n} (\tau d_j - \tau i_j) \tag{2}$$

Where $\bar{\tau}_{fd}$ is the average fault detection time, n is the total number of faults, τd_j is the time where j is the jth fault that has been detected τi_j denotes the time that each fault has been inserted.

3) **SDN Topology Aggregation**: The multi-domain cloud IoT device management requires to collect the SDN topology information from each of the domains at the cloud. To this end, the topology aggregation is required to form the global topology view. Therefore, we measure the total time required for aggregation, given by:

$$\tau_{agr} = \sum_{i=1}^{n} (\tau_i) \tag{3}$$

Where τ_{agr} is the average time to aggregate i number of topologies, which each of the topologies have τ time of completed aggregation. Equation (3) works for static topologies, however in a realistic scenario the topologies will change over time. Therefore, we also define the following:

$$\tau_{agr}(t) = \sum_{i=1}^{n} (\tau_i(t)) \tag{4}$$

In Equation (4) we calculate the total aggregation time for dynamic topologies that change over time. To accomplish this, we measure the aggregation time τ_{agr} as a function of the experimentation life time t.
Finally, the derivative of t_{agr} with respect to time indicates the rate of change of t_{agr}. Therefore in (5) we define:

$$\frac{\partial t_{agr}}{\partial t} = \lim_{\Delta t \to 0} \left(\frac{t_{agr}(t) - t_{agr}(t - \Delta t)}{\Delta t} \right) \tag{5}$$

Where in this work Δt is set to 1 second, which is a fair approximation since $t_{sampl} << \tau_{total}$ where t_{sampl} is the sampling time and τ_{total} is the total experiment time.

4) **Energy Consumption**: We measure the energy consumption of each individual process of SDIM, NETCONF Lite and LWM2M software, by performing energy monitoring per approach. The energy consumption consumed by each approach is monitored for one hundred minutes of experimentation. The monitoring is done by, implementing a script that stores the energy consumption of the container process over the time of the experiment. This allows to calculate the energy used throughout the provisioning period and the fault detection period and assess the potential impact that both may have on energy consumption.
5) **Total Device Energy Consumption** of all sensor nodes is also calculated during the experimentation period and is given in (6)

$$E(\tau) = \sum_{s} (P_s * \tau_s), S \in (Tx, Idle, DeepSleep) \tag{6}$$

Where $E(\tau)$ denotes the energy at time τ by measuring the sum of power consumption for different states such as Tx for transmit power $idle$ for idle mode and $DeepSleep$ for sleep mode. The currents used for measurements are provided by the manufacturer [53]

6) **Percentage of Network Control Messages** This metric evaluates how the control messages of all approaches impact the network. By measuring all control messages per action we are able to present results for all the devices during the experimentation. In (7) we present the model used to calculate the total number of control messages per approach.

$$T_{ctlm} = \sum_{i=0}^{n} (provm_i + faultsm_i + opsm_i + sdnm_i) \tag{7}$$

Where in (7) T_{ctlm} denotes the total control messages, measured by the sum for all control messages n where device provisioning control messages given as $provm_i$, $faultsm_i$ for fault detection control messages, for any device operations such as enable/disable control messages are defined as $opsm_i$ and finally for any other SDN related messages as $sdnm_i$, results to a total control messages summary.

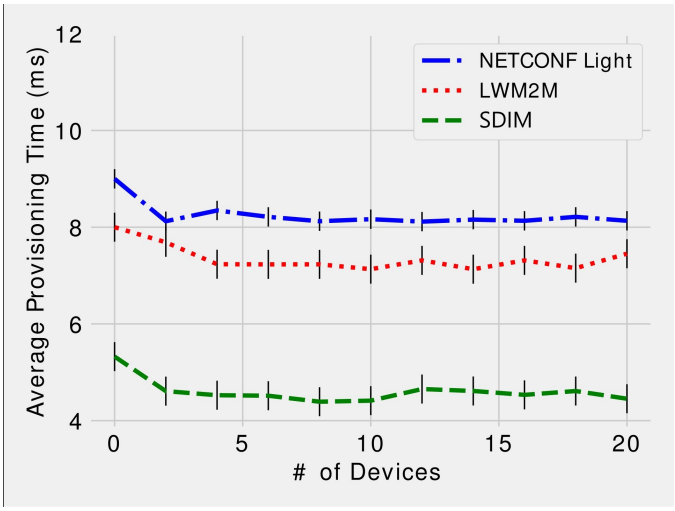
7) **CPU Performance**: For MEC nodes it is important to deploy light-weight functions, to this end CPU performance is observed throughout the fault detection period of one hundred minutes experimentation. Similarly to the energy consumption monitoring, we also deploy a script that stores the CPU over time.

8) **Completed Operation Time**: We define the time needed to enable or disable a device as "completed operation time". Note that during our experiments the devices can be controlled in following two ways:
   a) Enable devices by either increasing the accepted bandwidth or white listing a black listed device.
   b) Disable devices in the field

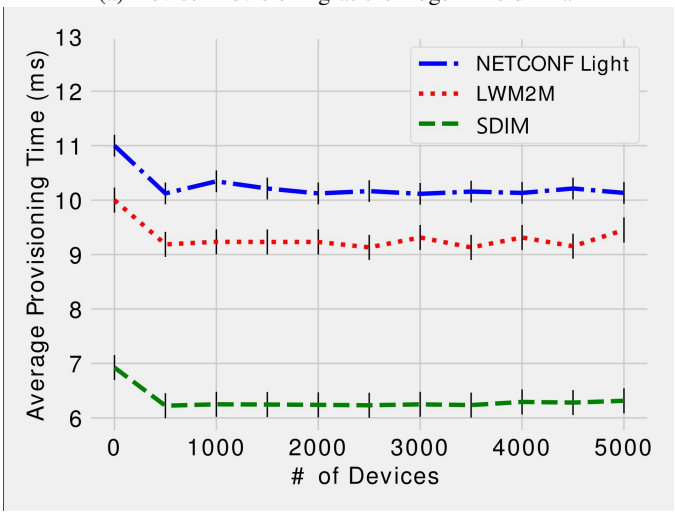
A. Edge and Cloud Device Provisioning

Figure 6 shows the average provisioning time for SDIM, LWM2M and NETCONF at the *edge* with respect to both field trial Figure 6(a) and emulated large trial networks Figure 6(b) per different number of devices. Note that both Figures report a time performance measurement for the special case of *zero devices*. These measurements, in particular, refer *only* to the time needed for system *initialization* and *not* to actual provisioning time. Initialization time is different for each framework, hence the different starting points in the corresponding Figures. Evidently, SDIM implies a much lower initialization time compared to the benchmarks for the same reasons it does so with respect to average device provisioning time, and which we explain next.

Following system initialization, both field trial and simulation results show that the average provisioning time for all approaches converges regardless of the increasing number of devices reported in the X-axis. Even more importantly, SDIM's averaged converged time performance (~4.5 ms in field trial; ~6.3 ms during emulation) significantly outperforms that of both NETCONF (~8.1 ms in field trial; ~10.1 ms during emulation) and LWM2M (~7.2 ms in field trial; ~9.2 ms during emulation). The former field trial performances denote an 80% and a 60% of overhead by NETCONF Lite and LWM2M, respectively, compared to SDIM. Likewise, the corresponding performance overheads under large network emulation are approximately 60.3% for NETCONF Lite and 46% for LWM2M, i.e. they are expected to be significantly high with respect to large network setups. This exhibited performance merit is due to SDIM's SDN nature, which enables to dynamically attach and configure new devices to the network through a *single* interface. Unlike that, both NETCONF Lite and LWM2M require the establishment of additional connectivity (i.e., ssh sessions) and imply an additional message burden to perform device provisioning.



(a) Device Provisioning at the Edge - Field Trial



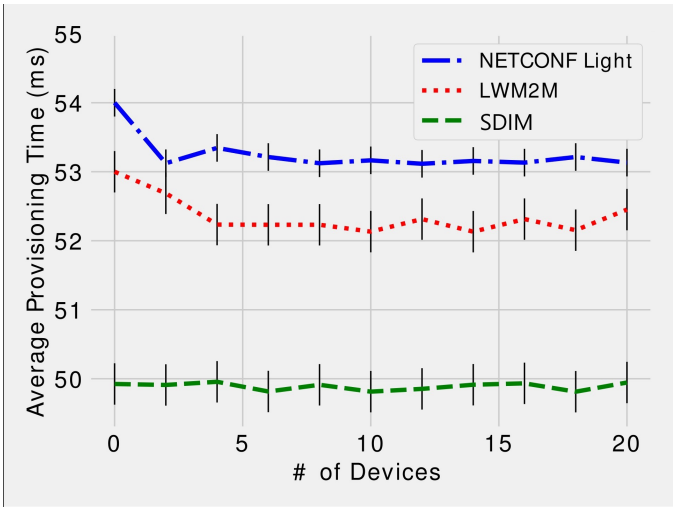
(b) Device Provisioning at the Edge - Emulation

Fig. 6: Device Provisioning Comparison of (a) Field Trial and (b) Emulated Large IoT Networks at the Edge. Note that the portrayed performance for zero devices refers to system initialization time and not to device provisioning time.

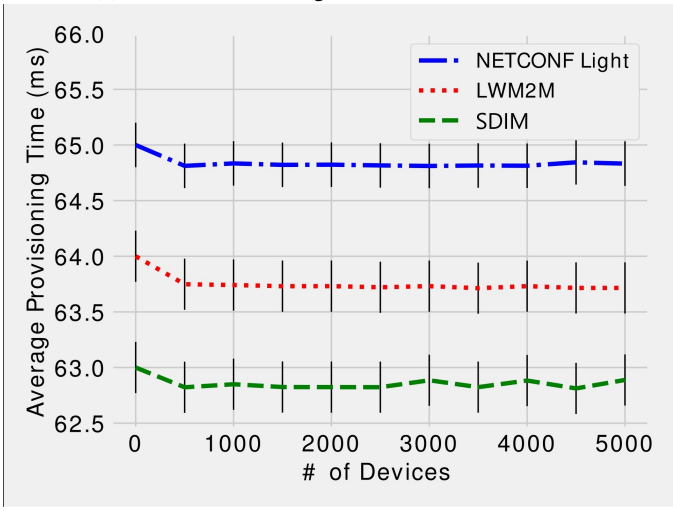
Moving on next to Figure 7, the figures show the average provisioning time for field trial (Figure 7(a)) and emulated large trial networks (Figure 7(b)) per different number of devices, this time with regard to *cloud* deployments of SDIM and the benchmark frameworks. But before we elaborate further into these performances, we remind the reader that

SDIM is *designed especially* for *edge deployments*. The results reported here for cloud deployments serve the purposes of a more holistic evaluation approach, where even the lower gains reported yield an added value to our SDN-based framework design.

As done in Figure 6, we report system initialization time and *not* device provisioning time for the special case of zero devices on the X-axis. In overall, the results are again favorable for SDIM. But as expected, the measurements on both system initialization as well as average provisioning times are higher for all frameworks in comparison to the edge deployment case of Figure 6. Morever, SDIM's time performance gains against NETCONF Lite and LWM2M are at a smaller scale compared to the edge deployment case. Specifically, the averaged converged time performance in the cloud for NETCONF Lite (~53.1 ms in field trial; ~64.8 ms during emulation) accounts for an approximate increase of 6.4% in field trial and 3.2% in the emulated large network, respectively, compared to SDIM (~49.9 ms in field trial; ~62.8 ms during emulation). A corresponding comparison of LWM2M (~52.2 ms in field trial; ~63.7 ms during emulation) against SDIM accounts again for an increase in average provisioning time by 4.6% and 1.4% with respect to the field trial and the large network emulation, respectively.



(a) Device Provisioning at the Cloud - Field Trial



(b) Device Provisioning at the Cloud - Emulation

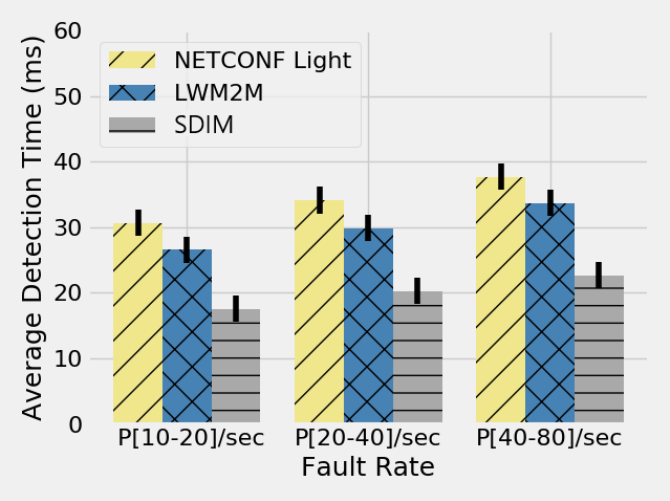
Fig. 7: Device Provisioning Comparison of (a) Field Trial and (b) Emulated Large IoT Networks at the Cloud. Again, the portrayed performance for zero devices refers to system initialization time and not to device provisioning time.

B. Edge and Cloud Fault Detection Evaluation

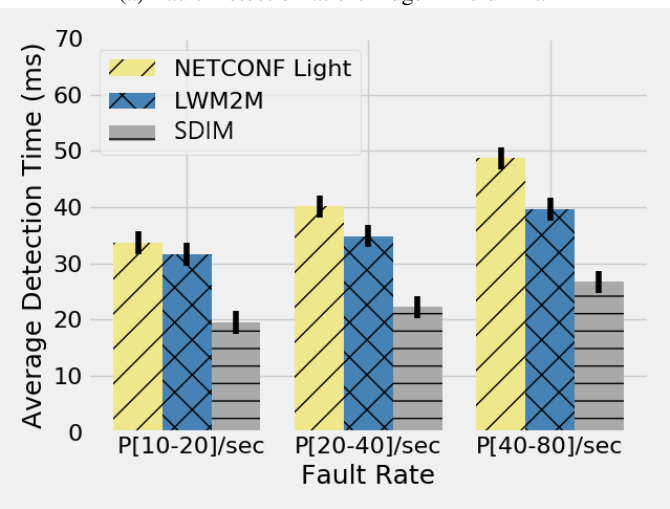
Figure 8a shows the average fault detection time during the fault insertions time that follows a Poisson distribution. The fault detection performance of all three approaches is presented and as shown in Figure 8a SDIM is faster at detecting faults for all of the experiments conducted. The SDIM implements the Algorithm 2 described in Section III for fault detection which decreases the complexity over time, therefore contributing to reduce the fault detection time. On the other hand, NETCONF Lite and LWM2M operate with configurations detection and message based fault monitoring. This way the detection of faults requires more time, since it depends on the devices availability. Figure 8b depicts results for fault detection, with emulated 5000 devices. We can see that the number of devices does not impact significantly the detection time, since the change of behaviour is very low.

Figure 9a shows the average fault detection time for field trial and emulated devices, with faults being inserted following a Poisson distribution. We compare SDIM, LWM2M and NETCONF, by deploying them in the cloud. Unsurprisingly, the fault detection time in the edge is lower than in the cloud. However, the lowest average detection time is still provided through the use of the SDIM. Additionally Figure 9a presents results for fault detection with emulated 5000 devices, still SDIM performs better. For the cloud fault detection experiments we found that fault detection time is being significantly affected by the cloud delay, in comparison with the edge network. The detection time is increased more than 10 times in all of the approaches. Additionally, although the SDIM SDN based fault detection is operating faster than the other benchmarks, here the difference is smaller than the edge by 15% to 20%. This is an interesting finding because it shows that the cloud detection is not leveraging SDN as much as fault detection in the edge of the networks.

Figure 10 presents results for 1 second of faults, for the worst case scenario of faults. This specific experiment inserts a total of 80 faults for 1000ms. Figure 10a zooms in the fault detected per approach and the detection time that we recorded for experiments in the edge of the network. We selected a three dimensional figure to depict that SDIM is performing better by not only detecting faster the faults, but as well detecting a larger amount of faults. The explanation for this is that SDIM is faster in detecting faults, therefore the total number of faults detected within a specific time is also higher. The same experiment is conducted in the cloud and the results are presented in Figure 10b. It is interesting to observe the larger detection time for all of the approaches. Again in the cloud the performance difference between SDIM and other benchmarks is smaller. Additionally, in the cloud all of the approaches are

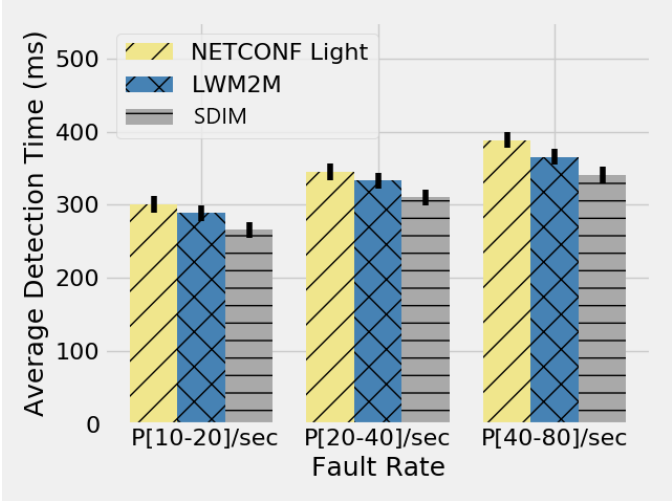


(a) Fault Detection at the Edge - Field Trial

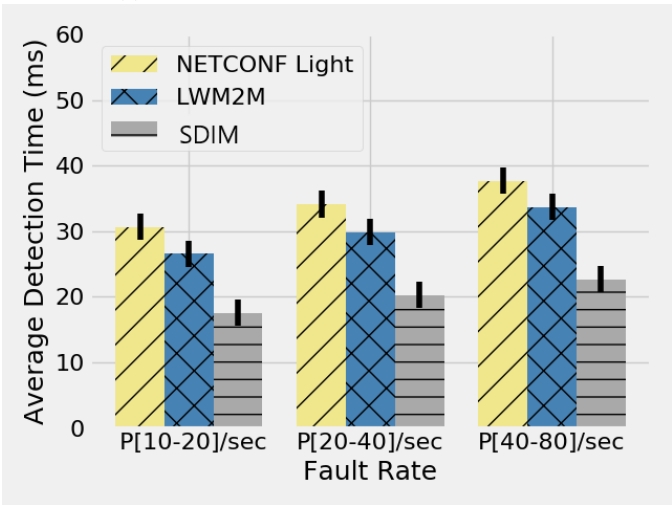


(b) Fault Detection at the Edge - Emulation

Fig. 8: Fault Detection Comparison of Field Trial and Emulated Large IoT Networks at the Edge



(a) Fault Detection at the Cloud - Field Trial



(b) Fault Detection at the Cloud - Emulation

Fig. 9: Fault Detection Comparison of Field Trial and Emulated Large IoT Networks at the Cloud

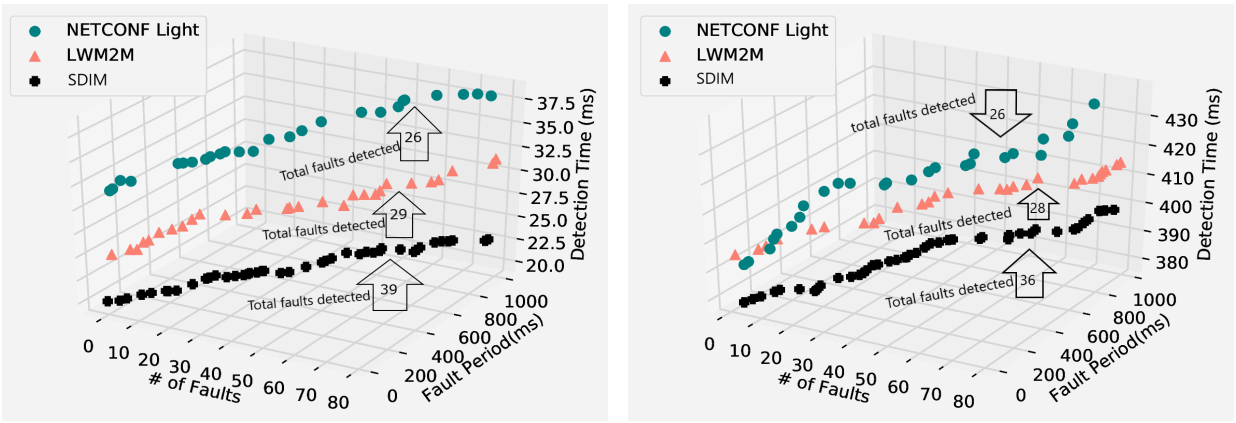
detecting almost 10% less faults. It is clear that cloud delay impacts significantly the detection process.

C. Topology Aggregation Performance

Figure 11 shows the elapsed time to complete SDN topology aggregation for different SDN topologies per different sets of access points (APs) for a constant number of devices equal to 1500 (see Figure 11a) and varied number of devices, i.e. 1500, 2000 and 2500, per APs (see Figure 11b). Furthermore, Figure 11 maps the equations (3), (4), and (5) to visualize the time of the static topology aggregation, dynamic topology aggregation and the rate of change. Additionally, we want to observe in real experiments how the overall SDN topology aggregation operates, and how it can be affected. As shown in Figure 11a, the modification of the topologies has a direct impact on τ_{agr}, as every new aggregation takes a larger time. This is due to the fact that the topologies change instantly, therefore the previous aggregation has not been completed yet. This results to a peak time at the 4th topology. After that point $\tau_{agr}(t)$ starts to drop. We observe the same behaviour also in Figure 11b, where the difference of devices per APs results into a small impact on the τ_{agr}, noting, however, that the behaviour of $\tau_{agr}(t)$ remains the same. Finally, the rate of change (5) allows to approximate the impact of dynamic topologies to the SDN cloud aggregation, highlighting the conclusions about the cloud SDN capabilities for IoT.

D. Energy Efficiency at Edge Nodes

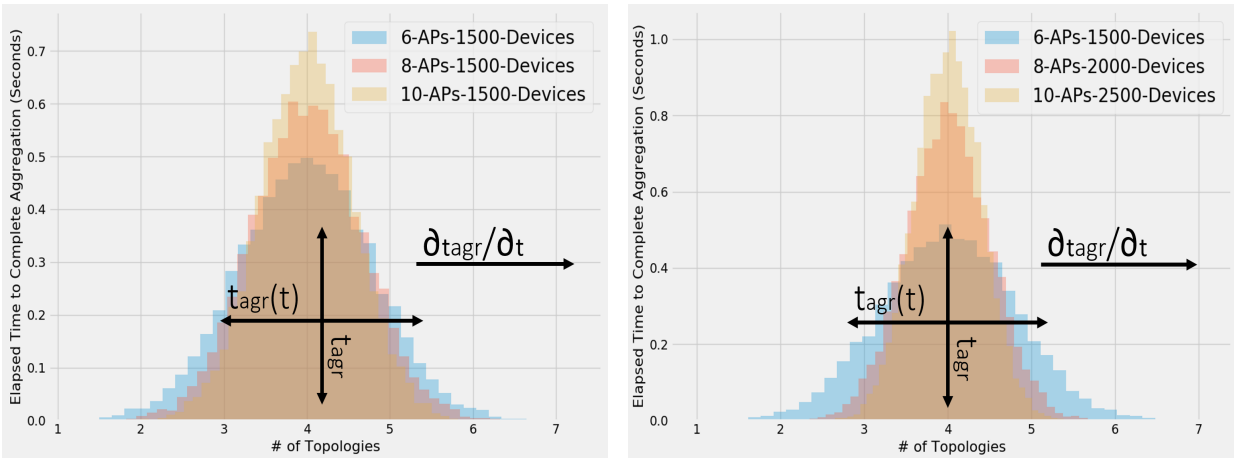
1) Energy Consumption During Provisioning Period: Figure 12a presents an energy consumption comparison among SDIM and the benchmark approaches during device provisioning. The Figure shows average consumption at all five MEC nodes in our setup after repeating device provisioning a hundred times, using 95% confidence intervals. Evidently, SDIM generally outperforms the other protocols. It needs on average 4.4% and 12% less energy compared to LWM2M and NETCONF light, respectively, with energy gains reaching at times as high as 20% and 15%, respectively. The reason



(a) Edge for 1 Second Fault Detection Speed

(b) Cloud for 1 Second Fault Detection Speed

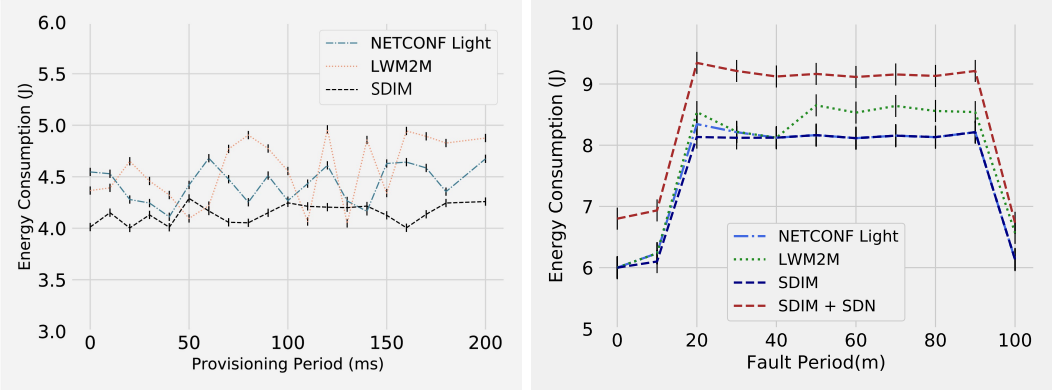
Fig. 10: Worst Case Faults Detection - 1 Second of Faults



(a) Constant # of Devices

(b) Varied # of Devices

Fig. 11: SDN Topology Aggregation for Constant Number of Devices (1500) and Different Number of Devices (1500, 2000, 2500) for Different Access Points (6, 8, 10)



(a) During provisioning period

(b) During fault detection

Fig. 12: Energy consumption at the edge

for these gains lies in the fact that SDIM offloads device and network topology discovery to the SDN controller before on boarding the devices to the platform. Therefore, SDIM performs very lightweight tasks during this period.

This finding is of *critical importance*, as IoT devices have generally limited processing capabilities and battery autonomy, thus leading to pushing much of the energy-hungry processing to the edge nodes. Nevertheless, edge nodes may be running

on battery themselves such as in Public Protection and Disaster Relief (PPDR) use case scenarios involving flying drones that form an ad-hoc network edge for controlling IoT devices and for continuously gathering and processing their sensory data.

2) Energy Consumption During Fault Detection Period: Figure 12b illustrates energy consumption for fault detection purposes during a hundred minute period of inserting faults to the network. The Figure includes both the aggregated average consumption of SDIM at the edge nodes and the centralised SDN controller (SDIM + SDN), as well as the "stand-alone" average consumption of SDIM at the MEC nodes against the benchmark protocols. The "stand-alone" consumption of SDIM is generally lower than LWM2M and practically identifies with the one exhibited by NETCONF light for most of the fault detection period in the Figure, converging into roughly 4.5-4.9% of lower energy needs compared to LWM2M. These results alongside the energy consumption levels during device provisioning verify that SDIM can yield significant energy gains, particularly in scenarios were the edge nodes work under energy constraints. Regarding the energy overhead of SDIM + SDN, this reveals a desired trade off achieved by SDIM by offloading tasks to a centralized SDN controller at the cost of increase energy consumption there where energy consumption is not an issue unlike the energy-restricted edge.

E. Network Control Messages

To provide a complete evaluation view including from a network perspective, we conduct experiments about network load referring to control messages only. We run the experiments for each approach and monitor the packets that are related to device provisioning, faults detection, device control (operations) and for SDIM also any SDN related messages. As depicted in Figure 14b SDIM is outperforming both LWM2M and NETCONF by over 20%, reducing significatly the control messages. It is important to note that this behaviour is expected since most of the information required by SDIM is collected during the network setup, hence we see increased control messages for the very beginning of the experiment.

F. CPU & Device Control Performance

1) CPU Performance During Fault Detection Period: We monitored and present in Figure 13a the CPU performance of SDIM, NETCONF light and LWM2M at the edge nodes during a period of one hundred minutes of sending data and conducting fault detection. All three approaches converge to a stable CPU consumption with SDIM (approx. 16%) and NETCONF light (approx 18.1%) doing so between the time period of (20, 90), and LWM2M (approx. 18.4%) between (45, 90). This shows that SDIM manages to also *reduce CPU consumption* at the edge by roughly 2.1-2.4%, which is one of the reasons alongside messaging reduction for reducing energy consumption at the edge with SDIM. For completeness, we include SDIM + SDN in our Figure, i.e. including CPU consumption at the SDN controller along with CPU consumption at the edge nodes. As with previous results regarding energy consumption, the evident CPU consumption overhead denotes the cost traded for leveraging the benefits of SDN including the ability of SDIM to move resource-hungry computational tasks away from the edge to a centralised point where we can assume that resources (either physical or virtual) are *not* scarce and inexpensive to find.

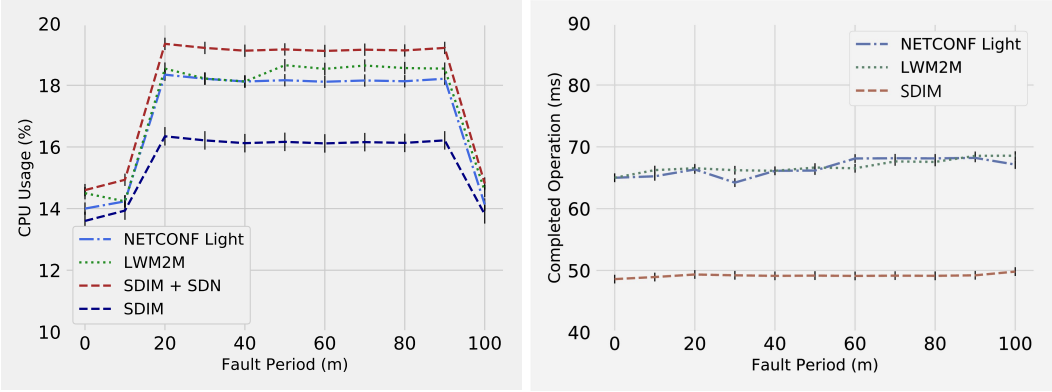
2) Control Operations: Based on the two control operations that define "Completed Operation time" (see point 8 on page 11), we evaluate the required time by all the framework approaches in order to successfully control the traffic of one or many devices. Note that we focus on how much time each operation requires to complete successfully an action over the devices. Figure 13b shows that SDIM performs significantly better than both LWM2M and NETCONF, whose performances tend to converge with only slight variations. Specifically, control operations time gains for SDIM span from 15 ms up to almost 19 ms, which translates to reduction levels between 23% up to 27%. This poses a powerful feature for SDIM, particularly for IoT use cases that involve quick control reactions such as in the context eHealth or industrial safety.

3) Sensor Energy Consumption Performance: To examine any potential impact on the sensor network energy consumption that the device management approaches might have, we measure the average energy consumption of the total number of devices. As illustrated in Figure 14a, there is *no* impact on sensors' energy consumption. In fact, SDIM yields a non-negligible *improvement* on energy consumption during our experiments that reaches up to 2.5%.

VI. Conclusion

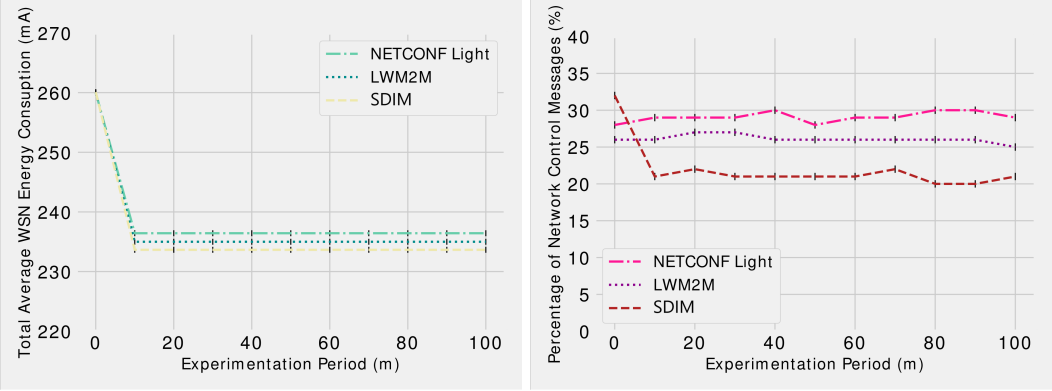
This work focuses on the scalable multi-domain IoT device management problem. We design, analyse and test a novel Software Defined IoT Management (SDIM) framework over a *real* IoT sensor network integrated with MEC capabilities. SDIM is designed for *edge* network deployments, targeting multi-domain Wireless Sensor Networks (WSNs) where cloud-based centralized device management can *not scale* well for dense IoT deployments. Nevertheless, SDIM may be also used for cloud-based monitoring and/or control over all IoT domains thanks to our implemented SDN Topology Aggregation (SDTA), which enables global network topology aggregation.

Based on performance metrics such as device provisioning time, operational fault detection time, energy and CPU performance of the utilized MEC nodes, we show that SDIM outperforms other state-of-the-art IoT management frameworks in both emulated large IoT networks and in real field trials. Specifically, SDIM can lower average device provisioning times by 60-80% compared to NETCONF Light and 46-60.3% compared to LWM2M. Also, SDIM can reduce the average operational fault detection time by approximately 33% compared to LWM2M and by 40% compared to NETCONF light. In further, SDIM reduces control operations time up to 27%, hence posing a *powerful feature* for use cases with *time-critical* control requirements such as eHealth. Last, SDIM manages to both reduce CPU consumption and to have important energy consumption gains at MEC nodes during device provisioning and operational fault detection periods relevant to the benchmark framework deployments.



(a) CPU Performance during Fault Detection Period (b) Control Time Required for Device Control

Fig. 13: CPU & Device Control Performance



(a) Energy Consumption of Devices (b) Total Network Load

Fig. 14: Energy & Control Messages Performance

Future work includes virtualizing the procedures of device management as Virtual Network Functions (VNFs) and evaluate the potential benefits of MANO orchestration for IoT device management. Also, we will consider expanding the scope of our field-trial experiments using an even larger real IoT network deployment. Finally, we intend to experiment with more tangible IoT use cases such as WSN on drones, wearables and Vehicle-to-everything (V2X) 802.11p networks.

Acknowledgment

This paper was performed under the REPLICATE and 5GinFIRE projects funded by the HORIZON 2020 the EU Framework Programme for Research and Innovation.

References

[1] U. Nations, "World's Population Increasingly Urban with More Than Half Living in Urban Areas," Tech. Rep., 2014. [Online]. Available: http://www.un.org/en/development/desa/news/population/world-urbanization-prospects-2014.html

[2] ©International Telecommunication Union (ITU-T) Focus Group on Smart Sustainable Cities, "Overview of Smart Sustainable Cities and the Role of Information and Communication Technologies (ICTs)," Tech. Rep., 2014.

[3] A. Gharaibeh, M. A. Salahuddin, S. J. Hussini, A. Khreishah, I. Khalil, M. Guizani, and A. I. Al-Fuqaha, "Smart Cities: A Survey on Data Management, Security, and Enabling Technologies," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2456–2501, 2017. [Online]. Available: https://doi.org/10.1109/COMST.2017.2736886

[4] A. Zanella, N. Bui, A. P. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014. [Online]. Available: https://doi.org/10.1109/JIOT.2014.2306328

[5] J. A. Stankovic, "Research Directions for the Internet of Things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, Feb 2014.

[6] A. I. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015. [Online]. Available: https://doi.org/10.1109/COMST.2015.2444095

[7] C. Mulligan and M. Olsson, "Architectural implications of smart city business models: an evolutionary perspective," *IEEE Communications Magazine*, vol. 51, no. 6, 2013. [Online]. Available: https://doi.org/10.1109/MCOM.2013.6525599

[8] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2018. [Online]. Available: https://doi.org/10.1109/JIOT.2017.2767608

[9] D. Evans, "Next generation networks – frameworks and functional architecture models: Overview of the internet of things," Tech. Rep., 2012.

[10] E. ©Internet Engineering Task Force (IETF)-M. Ersue, "An Overview of the IETF Network Management Standards," Tech. Rep., 2012.

[11] Z. Sheng, C. Mahapatra, C. Zhu, and V. C. M. Leung, "Recent advances in industrial wireless sensor networks toward efficient management in

IoT," *IEEE Access*, vol. 3, pp. 622–637, 2015.

[12] A. Sehgal, V. Perelman, S. Kuryla, and J. Schönwälder, "Management of resource constrained devices in the internet of things," *IEEE Communications Magazine*, vol. 50, no. 12, pp. 144–149, 2012. [Online]. Available: https://doi.org/10.1109/MCOM.2012.6384464

[13] D. Evans, "Open Mobile Alliances," Tech. Rep., June 2012.

[14] Open Mobile Alliances, "OMA Lightweight M2M," Tech. Rep. [Online]. Available: http://echnical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/oma-lightweightm2m

[15] F. V. D. Abeele, J. Hoebeke, I. Moerman, and P. Demeester, "Fine-grained management of CoAP interactions with constrained IoT devices," in *2014 IEEE Network Operations and Management Symposium, NOMS 2014, Krakow, Poland, May 5-9, 2014*, 2014, pp. 1–5. [Online]. Available: https://doi.org/10.1109/NOMS.2014.6838368

[16] M. Chiang and T. Zhang, "Fog and IoT: An Overview of Research Opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016. [Online]. Available: https://doi.org/10.1109/JIOT.2016.2584538

[17] Y. Jararweh, M. Al-Ayyoub, A. Darabseh, E. Benkhelifa, M. A. Vouk, and A. Rindos, "SDIoT: a software defined based internet of things framework," *J. Ambient Intelligence and Humanized Computing*, vol. 6, no. 4, pp. 453–461, 2015. [Online]. Available: https://doi.org/10.1007/s12652-015-0290-y

[18] D. M. F. Mattos, O. C. M. B. Duarte, and G. Pujolle, "Profiling software defined networks for dynamic distributed-controller provisioning," in *7th International Conference on the Network of the Future, NOF 2016, Búzios, Brazil, November 16-18, 2016*, 2016, pp. 1–5. [Online]. Available: https://doi.org/10.1109/NOF.2016.7810139

[19] W. F. Elsadek and M. N. Mikhail, "Inter-domain Mobility Management Using SDN for Residential/Enterprise Real Time Services," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, Aug 2016, pp. 43–50.

[20] S. Bera, S. Misra, and A. V. Vasilakos, "Software-Defined Networking for Internet of Things: A Survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, 2017. [Online]. Available: https://doi.org/10.1109/JIOT.2017.2746186

[21] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013. [Online]. Available: https://doi.org/10.1109/MCOM.2013.6461195

[22] D. M. F. Mattos, O. C. M. B. Duarte, and G. Pujolle, "A resilient distributed controller for software defined networking," in *2016 IEEE International Conference on Communications, ICC 2016, Kuala Lumpur, Malaysia, May 22-27, 2016*, 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICC.2016.7511032

[23] A. Mavromatis, A. P. Da Silva, K. Kondepu, D. Gkounis, R. Nejabati, and D. Simeonidou, "A software defined device provisioning framework facilitating scalability in internet of things," in *2018 IEEE 5G World Forum (5GWF)*, July 2018, pp. 446–451.

[24] "Open Mobile Alliance. Lightweight Machine to Machine Technical Specification. Approved Version 1.0.1," Tech. Rep., July 2017.

[25] J. S. V. Perelman, M. Ersue and K. Watsen, *Network Configuration Protocol Light (NETCONF Light)*. Freemont, CA, USA: Internet Eng. Task Force (IETF), 2014.

[26] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, Feb 2018.

[27] L. Gürgen and S. Honiden, "Management of Networked Sensing Devices," in *MDM 2009, Tenth International Conference on Mobile Data Management, Taipei, Taiwan, 18-20 May 2009*, 2009, pp. 502–507. [Online]. Available: https://doi.org/10.1109/MDM.2009.88

[28] Z. Sheng, S. Yang, Y. Yu, A. V. Vasilakos, J. A. Mccann, and K. K. Leung, "A survey on the IETF protocol suite for the internet of things: standards, challenges, and opportunities," *IEEE Wireless Communications*, vol. 20, no. 6, pp. 91–98, December 2013.

[29] A. H. H. Ngu, M. A. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "IoT Middleware: A Survey on Issues and Enabling Technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017. [Online]. Available: https://doi.org/10.1109/JIOT.2016.2615180

[30] W. Cerroni, C. Buratti, S. Cerboni, G. Davoli, C. Contoli, F. Foresta, F. Callegati, and R. Verdone, "Intent-based management and orchestration of heterogeneous openflow/iot sdn domains," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, July 2017, pp. 1–9.

[31] J. de C. Silva, J. J. P. C. Rodrigues, J. Al-Muhtadi, R. A. L. Rabêlo, and V. Furtado, "Management platforms and protocols for internet of things: A survey," *Sensors*, vol. 19, no. 3, p. 676, 2019. [Online]. Available: https://doi.org/10.3390/s19030676

[32] B. Forum, "CPE WAN Management Protocol," Tech. Rep., 2014. [Online]. Available: http://www.broadband-forum.org/technical/download/

[33] C. Zhou and X. Zhang, "Toward the Internet of Things application and management: A practical approach," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2014, Sydney, Australia, June 19, 2014*, 2014, pp. 1–6. [Online]. Available: https://doi.org/10.1109/WoWMoM.2014.6918928

[34] Z. Su, Q. He, J. Zhang, and H. Li, "Research of Single Sign-On in Mobile RFID Middleware Based on Dynamic Tokens and WMMP," in *16th IEEE International Conference on Computational Science and Engineering, CSE 2013, December 3-5, 2013, Sydney, Australia*, 2013, pp. 1191–1194. [Online]. Available: https://doi.org/10.1109/CSE.2013.177

[35] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation Rate Maximization in UAV-Enabled Wireless-Powered Mobile-Edge Computing Systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927–1941, 2018. [Online]. Available: https://doi.org/10.1109/JSAC.2018.2864426

[36] L. L. de Souza, P. H. M. Pereira, J. de C. Silva, C. N. M. Marins, G. A. B. Marcondes, and J. J. P. C. Rodrigues, "IoT 5G-UDN Protocols: Practical Model and Evaluation," in *2018 IEEE International Conference on Communications Workshops, ICC Workshops 2018, Kansas City, MO, USA, May 20-24, 2018*, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICCW.2018.8403543

[37] J. de C. Silva, P. H. M. Pereira, L. L. de Souza, C. N. M. Marins, G. A. B. Marcondes, and J. J. P. C. Rodrigues, "Performance Evaluation of IoT Network Management Platforms," in *2018 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2018, Bangalore, India, September 19-22, 2018*, 2018, pp. 259–265. [Online]. Available: https://doi.org/10.1109/ICACCI.2018.8554364

[38] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," Tech. Rep., 2011.

[39] D. Kreutz, F. M. V. Ramos, P. J. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015. [Online]. Available: https://doi.org/10.1109/JPROC.2014.2371999

[40] S. Chakrabarty and D. W. Engels, "A secure IoT architecture for Smart Cities," in *13th IEEE Annual Consumer Communications & Networking Conference, CCNC 2016, Las Vegas, NV, USA, January 9-12, 2016*, 2016, pp. 812–813. [Online]. Available: https://doi.org/10.1109/CCNC.2016.7444889

[41] R. Muñoz, R. Vilalta, N. Yoshikane, R. Casellas, R. Martínez, T. Tsuritani, and I. Morita, "Integration of IoT, Transport SDN, and Edge/Cloud Computing for Dynamic Distribution of IoT Analytics and Efficient Use of Network Resources," *Journal of Lightwave Technology*, vol. 36, no. 7, pp. 1420–1428, April 2018.

[42] A. Mckeown, H. Rashvand, T. Wilcox, and P. Thomas, "Priority SDN Controlled Integrated Wireless and Powerline Wired for Smart-Home Internet of Things," in *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), Beijing, China, August 10-14, 2015*, 2015, pp. 1825–1830. [Online]. Available: https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.331

[43] P. Bellavista, C. Giannelli, T. Lagkas, and P. G. Sarigiannidis, "Quality management of surveillance multimedia streams via federated SDN controllers in fiwi-iot integrated deployment environments," *IEEE Access*, vol. 6, pp. 21 324–21 341, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2822401

[44] X. Li, P. Djukic, and H. Zhang, "Zoning for hierarchical network optimization in software defined networks," in *2014 IEEE Network Operations and Management Symposium, NOMS 2014, Krakow, Poland, May 5-9, 2014*, 2014, pp. 1–8. [Online]. Available: https://doi.org/10.1109/NOMS.2014.6838414

[45] N. Nikaein, X. Vasilakos, and A. Huang, "LL-MEC: enabling low latency edge applications," in *7th IEEE International Conference on Cloud Networking, CloudNet 2018, Tokyo, Japan, October 22-24, 2018*, 2018, pp. 1–7. [Online]. Available: https://doi.org/10.1109/CloudNet.2018.8549500

[46] A. Ksentini, M. Bagaa, T. Taleb, and I. Balasingham, "On using bargaining game for optimal placement of SDN controllers," in

2016 IEEE International Conference on Communications, ICC 2016, Kuala Lumpur, Malaysia, May 22-27, 2016, 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICC.2016.7511136

[47] F. Chen, C. Wu, X. Hong, Z. Lu, Z. Wang, and C. Lin, "Engineering traffic uncertainty in the openflow data plane," in *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*, 2016, pp. 1–9. [Online]. Available: https://doi.org/10.1109/INFOCOM.2016.7524612

[48] P. K. Sharma, S. Rathore, Y. Jeong, and J. H. Park, "SoftEdgeNet: SDN Based Energy-Efficient Distributed Network Architecture for Edge Computing," *IEEE Communications Magazine*, vol. 56, no. 12, pp. 104–111, 2018. [Online]. Available: https://doi.org/10.1109/MCOM.2018.1700822

[49] M. Uddin, S. Mukherjee, H. Chang, and T. V. Lakshman, "Sdn-based multi-protocol edge switching for iot service automation," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2775–2786, 2018. [Online]. Available: https://doi.org/10.1109/JSAC.2018.2871325

[50] Q. Shafi, A. Basit, S. B. Qaisar, A. Koay, and I. Welch, "Fog-Assisted SDN Controlled Framework for Enduring Anomaly Detection in an IoT Network," *IEEE Access*, vol. 6, pp. 73 713–73 723, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2884293

[51] H. Sun, F. Zhou, and R. Q. Hu, "Joint Offloading and Computation Energy Efficiency Maximization in a Mobile Edge Computing System," *IEEE Trans. Vehicular Technology*, vol. 68, no. 3, pp. 3052–3056, 2019. [Online]. Available: https://doi.org/10.1109/TVT.2019.2893094

[52] C. R. E. R. Ramon dos Reis Fontes, "Mininet-wifi," Tech. Rep., 2018. [Online]. Available: https://github.com/intrig-unicamp/mininet-wifi

[53] Pycom, "Lopy datasheet," Tech. Rep., 2018. [Online]. Available: https://www.mouser.com/datasheet/2/872/lopy-specsheet-1129426.pdf